SSD-TR-91-24

AEROSPACE REPORT NO.
TR-0091(6904-25)-1

Contractor Software Engineering Environment Evaluations Summary Report

Prepared by

RANWA HADDAD and HOMA TARAJI
Computer Systems Division
Engineering and Technology Group

May 1991

Prepared for

SPACE SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
Los Angeles Air Force Base
P. O. Box 92960
Los Angeles, CA 90009-2960

Programs Group

THE AEROSPACE CORPORATION
El Segundo, California

This final report was prepared by the Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-88-C-0089 with the Space Systems Division, P.O. Box 92960, Worldway Postal Center, Los Angeles, California 90009-2960. It was reviewed for The Aerospace Corporation by W. K. Clarkson, Principal Director, Software Engineering Subdivision, Computer Systems Division. The Air Force Computer Engineer was M. Zambrana.

This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

M. Zambrana
Computer Engineer
USAF SSD/SDEC

Lt Col L. Kwasigroh
Chief, Computer Resources Division
USAF SSD/SDEC

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approval for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) TR-0091(6904-25)-1	5 MONITORING ORGANIZATION REPORT NUMBER(S) SSD-TR-91-24
6a NAME OF PERFORMING ORGANIZATION The Aerospace Corporation 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Space Systems Division Air Force Systems Command
6c ADDRESS (City, State, and ZIP Code) 2350 E. El Segundo Blvd. El Segundo, CA 90245	7b ADDRESS (City, State, and ZIP Code) Los Angeles Air Force Base P. O. Box 92960 LosAngeles, CA 90009-2960
8a NAME OF FUNDING/SPONSORING ORGANIZATION Space Systems Division Air Force Systems Command 8b OFFICE SYMBOL (If applicable) SDE	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) Los Angeles Air Force Base P. O. Box 92960 Los Angeles, CA 90009-2960	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO. / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Contractor Software Engineering Environment Evaluations Summary Report

12 PERSONAL AUTHOR(S)
Ranwa Haddad and Homa Taraji

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM Jan. 1990 TO Dec. 1990	May 1991	31

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Software Engineering Environments, SEE, ISEE, Tools, Software Development Tools, Software Development Environments

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes the results of a task to evaluate Software Engineering Environments (SEE) at defense contractors. Ten major contractors were contacted. All of them currently are involved in SEE developments, although at different levels. Some still are in the early definition phases, while others have a SEE in place. Four of them were evaluated. The SEEs were evaluated along three different dimensions: their functionality, their architecture, and their usage. All four SEEs support Ada and offer training courses. However, the SEEs are weak in their support of testing functions. The four evaluated SEEs are in a state of evolution. Although SEEs can potentially be important factors in quality and productivity improvements, and DOD-STD-2167A mandates the use of a SEE, building a SEE is a costly and lengthy process. Therefore, a methodology is needed for evaluating contractor SEEs and for determining the cost/benefit tradeoffs. This task is the first step towards defining a SEE evaluation methodology for SDD contracts.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
M. Zambrana/Lt. Col L. Kwasigroh		SDEC

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

EXECUTIVE SUMMARY

This report summarizes the results of a task to evaluate Software Engineering Environments (SEE) at defense contractors. Ten major contractors were contacted. All of them are currently involved in SEE developments, although at different levels. Some are still in the early definition phases, while others have a SEE in place. Four SEEs were evaluated. The evaluations generated valuable information concerning both the evaluated SEEs and the process of evaluating SEEs. The SEEs were evaluated along three different dimensions: their functionality, their architecture, and their usage.

Initial findings from the study were as follows:

- A number of DOD contractors do not currently have a SEE in place.
- The Ada language is supported by all the evaluated SEEs.
- Defining and building a SEE is a very lengthy process.
- The four evaluated SEEs are in a state of constant evolution.
- All four SEEs are weak in their support of testing functions.
- Some SEE usage data are being collected but not very systematically.
- Every one of the evaluated SEEs offered training courses.
- Strong barriers exist between SEE developers and SEE users.
- The investment of large sums of money into developing a SEE puts pressure on the contractor's management to capitalize on their investment.

Although SEEs can potentially be important factors in quality and productivity improvements, building them is a costly and lengthy process. DOD-STD-2167A mandates the use of a SEE, yet no direction exists for how to acquire a SEE and how to optimize it for a given project. Because of this, more efforts need to be expended in evaluating the SEEs and in analyzing the potential benefits derived from the SEEs. Guidelines need to be established for evaluating SEEs, and SEE usage metrics need to be defined and collected.

CONTENTS

1. PURPOSE OF THE REPORT

The purpose of this report is to summarize a study performed of Software Engineering Environments (SEEs) currently available at government contractors.

This report describes the study, chronicles its activities, and summarizes its results. Project objectives are described in Section 2. Section 3 contains definitions of terms that are used in the report, while Section 4 describes the details of the activities that were performed. SEE findings are summarized in Section 5, and Section 6 discusses some of the lessons learned about the SEE evaluation process.

2. PROJECT OBJECTIVES

The standard DOD-STD-2167A mandates the use of systematic and well-documented software development methods to perform the different activities of the software life cycle phases. It also calls for establishing SEEs for performing the software engineering tasks on Department of Defense (DOD) software projects. As a result of this mandate and as the need for managing the cost of software projects has increased, there has been a significant growth in the number of SEEs that contractors have developed and used on software projects. Contractors decided to look to SEEs for ways to reduce their software costs and improve their software quality.

As SEEs grow in number and diversity, a mechanism for the evaluation of SEEs on DOD contracts is essential to a fair assessment of contractors' ability to develop and/or deliver appropriate SEEs. Furthermore, the government needs to further monitor the development of SEEs. Often a contractor's SEE is turned over to the government in the maintenance phase. For these purposes, the government should use proper guidelines or develop a methodology to evaluate proposed SEEs in source selection.

The SEE Evaluation Methodology project was initiated at The Aerospace Corporation in order to define a methodology for evaluating SEEs in source selection and for monitoring SEEs during system development and maintenance. The first task in this project was Contractor SEE Evaluations, which is the subject of this report. The Contractor SEE Evaluations task was to develop knowledge of existing contractor SEEs, to gain insight into the main issues that currently face DOD contractors who are trying to establish SEEs, and to gain some experience in SEE evaluations.

3. DEFINITIONS

Software Engineering is a discipline that has been subject to a significant proliferation of terms and concepts. New words are created at major conferences and technical meetings all the time. To avoid ambiguities, a set of software engineering terminologies that are used in this report are defined below.

A Software Engineering Environment (SEE) is an environment that automates the different activities of the software life cycle through an integrated collection of software development tools and methodologies. The number of life cycle activities that are automated by a SEE and the level of the automation vary depending on the SEE. As a minimum, every phase of the software development life cycle, as well as Configuration Management and Document Production, should be supported to some extent.

SEE developments have been greatly influenced by the practice of computer-aided software engineering (CASE). CASE is the widespread application of computer technology to software engineering techniques. In recent years, CASE has become a key concept for improving software development productivity and reducing software cost. With the increasing usage and diversity of CASE tools, the problem of tools interoperability has developed. Tools often do not interface with each other, and different CASE tools are used for different phases of the software life cycle. Users must learn multiple interfaces and put in extra effort to move between tools and between different phases of the software life cycle.

A SEE provides a unified support mechanism across the software lifecycle phases, in which CASE tools are one of several components. SEE components can be looked at from two points of view: functional and architectural. From a functional standpoint, the SEE components are its capabilities, and they can be mapped to the different tasks performed in the software development life-cycle, such as requirements definition, requirements analysis, preliminary design, and so forth, as defined in DOD-STD-2167A.

The second view of the SEE components is the physical or architectural view. Several attempts have been made at creating a standard reference model to illustrate the different physical components of SEEs. Of these attempts, the one that seems to be more widely accepted is the "toaster model." This model, shown in Figure 1, was first presented at the CASE '89 Standards Coordination Meeting by George Tatge of Hewlett Packard Company. The main components in this model are the tools, the user interface, a repository, task management services, and data integration services. The latter two are integration layers between the tools and

the user interface on one side and the repository on the other. A full description of the toaster model is given by D. Mularz[1].

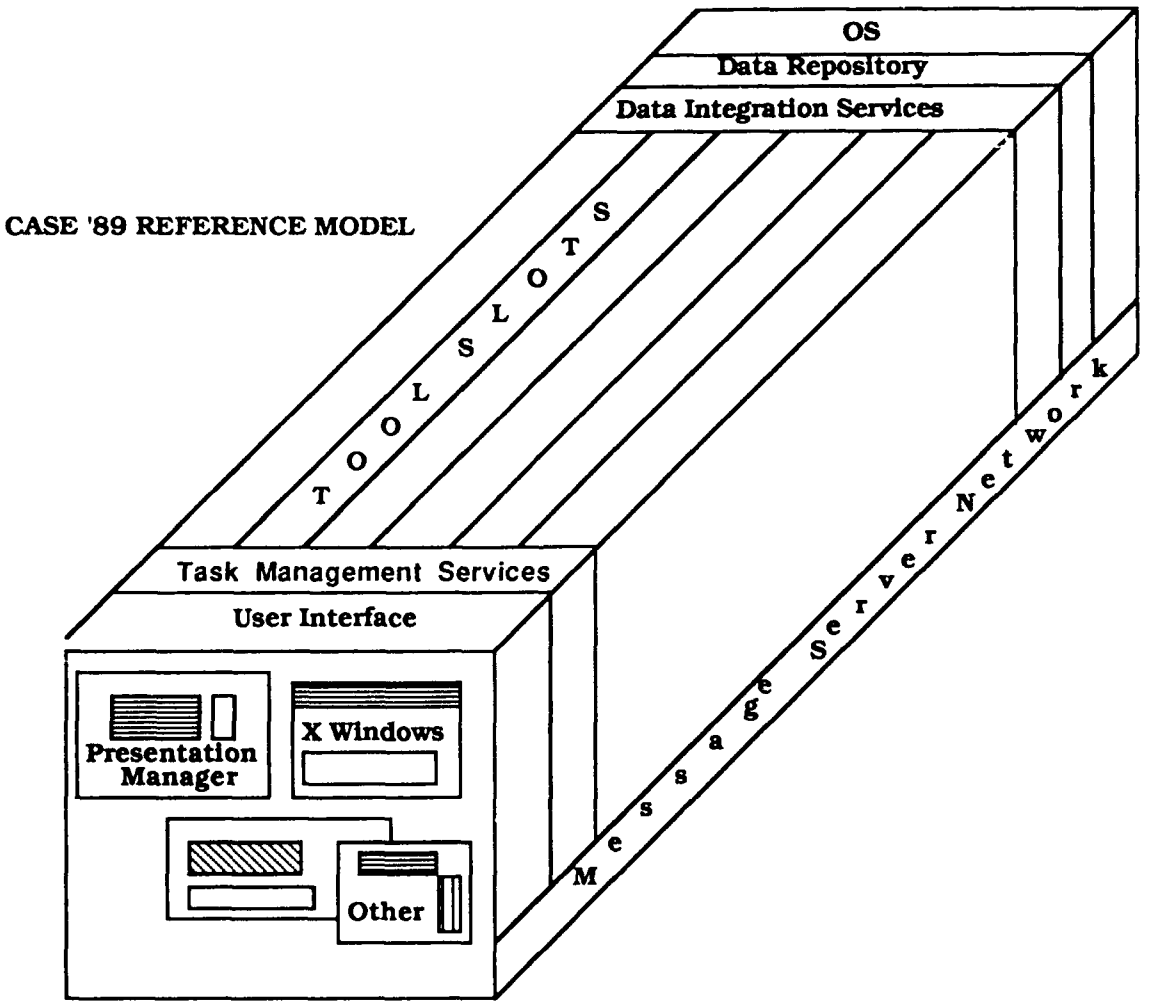

Figure 1. The Toaster Model, a SEE Reference Model

[1]Mularz, D., private communications (a paper being published at the MITRE Corporation).

Repositories are the basis for integration in SEEs. A Repository is a mechanism for defining, storing, and managing all information and objects pertaining to the development and maintenance of a software system. Objects such as process models, data models, design models, code modules, documents, relationships, actions, and so forth are captured in the Repository and made available to the different tools in the SEE.

Standards facilitate the exchange of information in a SEE. This exchange of information may occur between the users and the SEE (user interface), the tools in the SEE (tools interface), and the data in the repository (data interface).

In a software development environment, a user is faced with a collection of tools, each of which has its own interface. Although it is extremely nonproductive for users to spend time learning and using different interfaces, graphical user interface (GUI) standards are still limited to individual environments. Many GUIs (Motif, DecWindows, Windows 3.0, MacMultifinder, X Windows System, and so forth) have emerged in recent years, and they all seem to have been equally accepted as industry standards within a given environment.

Commercial tools and in-house tools are often used together in a SEE to provide various functionalities. These tools need to exchange data through a standard language. Several standards groups are working to develop standards for exchange of information between CASE tools; for example, CDIF (CASE Data Interchange Format) and CAIS (Common APSE[2] Interface Set).

Efforts are also under way for developing standards for data interface. For example, IRDS (Information Resource Dictionary System) provides standards for data sharing among repositories in addition to managing the recording, storing, and processing of data descriptions within a single repository.

Tailorability refers to the ability of a SEE to be modified in order to support different projects' needs. Once a SEE is established, which usually is based on a specific project's needs, other projects may want to benefit from its capabilities. In other words, SEEs must be flexible and easily customized to the needs of various projects. Tailorability is a major factor in justifying the high up-front cost of SEE developments.

[2] Ada Programming Support Environment

Reusability is an approach to software development in which software is not developed from scratch. Reusability in SEE development applies to the reuse of all or some of the components of the SEE. Reusability implies that SEE components must be developed with the intention of being reused, as is or modified, on other projects. Like tailorability, reusability is an important factor for justifying the high up-front cost of SEE developments.

4. TASK DESCRIPTION

To develop expertise in evaluating SEEs, a study of DOD contractor SEEs was conducted. The study was in the form of a series of SEE evaluations. First, three items needed to be identified: which contractors to select for the study, how to evaluate them, and what other studies exist that could be used in the evaluation.

Contractor selection was based on contractors' history in SEE development, their affiliation with Space Systems Division, and their willingness to participate in the study. Several of the major DOD contractors were contacted in the selection process which resulted in the selection of four SEEs as described in Section 4.1.

The evaluations had to be based on contractor information provided at the contractor's discretion. Therefore, the evaluations were based on documents provided by the contractors and, for two of the SEEs, a demonstration of the SEE and an interview with the SEE developers, driven by a questionnaire which had been prepared in advance. Most of the information gathering was performed between November 1989 and May 1990. The evaluation process is further described in Section 4.2.

A number of related studies, such as tool evaluations, SEE specifications, and so forth were reviewed; they are described in Section 4.3. However, none of these studies address the issue of evaluating an entire SEE.

4.1. Contractor Selection

Based on the authors' knowledge of on-going SEE projects, a list was prepared of the contractors involved in developing or using a SEE. A phone screening showed that many of these contractors are still in the early phases of defining their SEE. Furthermore, some of the contractors who had a SEE in place were not interested in sharing their SEE information, for a variety of reasons. Some did not have the time, since this activity would not be chargeable to a project they were working on. Others were concerned about being judged by a government agency based on the results of this study. A group expressed concern over the proprietary nature of their tools and methods. On the other hand, some of the contractors were very open and eager to share their SEE information.

Altogether, ten contractors were contacted, all of which were involved with SEE developments at one level or another. Four of the contractors were still in the early stages of planning their SEEs, and two other contractors were unable to participate in the study. The remaining four are the subject of the evaluations. Some of the information acquired regarding the contractors' SEEs is proprietary; therefore, no contractors

are mentioned by name in this report, and the four evaluated SEEs are referred to as SEE1, SEE2, SEE3, and SEE4.

4.2. Evaluation Process

In general, the process consisted of contacting contractor personnel who were knowledgeable about the SEE being evaluated, and soliciting information about their SEE through a site visit and/or existing documentation. The preference was to visit the SEE on site, but that was not always possible due to funding constraints on the contractor's part. On-site demonstrations were attended for two of the SEEs: SEE1 and SEE2, and the evaluations of SEE3 and SEE4 were based on available documentation and phone calls. The information gathering was driven by a predefined questionnaire. The questionnaire was designed to capture information about the architecture of the SEE, the capabilities of the SEE, the process of building the SEE, and the process of using it. A copy of the questionnaire can be found in the Appendix. It was developed after a careful review of related studies, which are described in Subsection 4.3.

All contact points were very gracious and generous with their time, especially considering that they had no existing contract with the Space Systems Division that was applicable to SEE evaluations. Numerous attempts were also made at contacting contractor personnel who had used the SEEs being evaluated; however, none of the users were able to spare any time for sharing the information. Some usage data were obtained from the SEE developers nonetheless, but they were not confirmed with the users.

4.3. Related Studies

A literature search on works relating to the evaluation of SEEs at military contractors uncovered a number of different studies. All the studies described below were reviewed, and their results were used in formulating an evaluation approach and a SEE questionnaire.

Most of the studies address the evaluation of specific software tools. A study by the Software Engineering Institute describes how to evaluate and select Software Engineering tools (See Reference 1). The Software Technology Support Center has defined a methodology called STEM for classifying, selecting, and evaluating software development tools (See Reference 2). MITRE is studying the current practices in the use of CASE tools (See Footnote 1). MITRE has also performed a number of evaluations of existing CASE tools (See References 3 and 4).

One DOD standard exists for the acquisition of SEEs. "Software Support Environments," DOD-STD-1467," and the accompanying handbook MIL-HDBK-782, "Military Handbook for Software Support Environment Acquisition," provide guidelines for acquiring a Support Environment for software delivered to the DOD. The standard and its accompanying handbook define a step-by-step approach for the government to ensure that a complete life-cycle support capability is acquired for contractually deliverable software. The standard and its handbook, however, do not distinguish between a support environment that is a collection of tools and SEEs as they are currently defined and where a certain amount of cohesion is expected between the tools, nor do they provide any guidance for evaluating SEEs beyond ensuring that they provide end-to-end tool support.

5. SEE FINDINGS

This task produced useful information about different aspects of SEEs. Significant information was acquired about the structure of the evaluated SEEs, their architecture, their components, what tools they comprise, and which methodologies they support. Of equal importance was the information acquired about the processes followed in establishing and using the evaluated SEEs, such as how the SEEs are funded, how they are promoted, and so forth. General findings about the four SEEs are summarized in Subsection 5.1, and a comparison of the SEEs' architectural features is presented in Subsection 5.2.

5.1. General Findings

The following general findings were derived from a sample of ten surveyed contractors of which four SEEs were evaluated. This is a relatively small sample which may not represent all the existing DOD contractor SEEs. The significance of the findings, however, lies in the fact that the companies that built the SEEs are four of the major DOD contractors and that they have established a reputation in developing software support environments.

- A number of DOD contractors do not currently have a SEE in place.

- Defining and building a SEE is a very lengthy process. Except for SEE1 which was not completed at the time of the evaluation, the other three evaluated SEEs had evolved over a period of more than five years.

- The four SEEs are in a state of constant evolution. Each one has evolved from some previous SEE, and they all have long lists of planned improvements.

- The SEEs are research and development funded, probably because of the cost of development. No cost data were obtained from the evaluations; however, figures in excess of $200 million were given informally at a recent STARS[3] workshop for one of the evaluated SEEs as well as another SEE that is not part of this study. Three of the evaluated SEEs were developed over periods in excess of five years

[3] STARS (Software Technology for Adaptable, Reliable Systems) is a DARPA (Defense Advanced Research Projects Agency) program which is defining a framework for SEEs that can be populated with commercial tools, can be adapted to different platforms, and will facilitate the utilization of reusable components. The STARS SEE framework would reduce the cost of developing SEEs by facilitating the reuse of existing SEE components and of third party software. STARS is being coordinated with academia, industry, and DOD contractors but is still in early development stages.

and by large development teams. Undoubtedly they incurred high labor costs.

• The Ada language is supported by all the evaluated SEEs; in the case of one SEE, it is a requirement for using the SEE.

• Although the goal of all the evaluated SEEs is to provide end-to-end software life cycle support, only three of them currently do so, and each has different areas of strengths and weaknesses.

• All four SEEs are weak in their support of testing functions. Of all the software development and management functions, testing seemed to be the most uniformly neglected. None of the evaluated SEEs had any code analysis or test coverage tools, and only one SEE supported traceability between tests and requirements.

• None of the evaluated SEEs have tailorability features such as user-definable database attributes, conditionally triggered procedures, project specific reports, and so forth.

• There are some data on the usage of the SEEs but they are insufficient. One contractor reported productivity gains of 54% and quality gains of 350% on one project based on Lines Of Code per Man Hour (LOC/MHR) and Software Change Reports per KLOC (Thousand Lines of Code). Another contractor reported a 50% decrease in bid rates due to the SEE usage, also based on Lines Of Code per Man Month (LOC/MM) and Detected Errors per KLOC. The LOC/MM and Detected Errors per KLOC are the only SEE usage metrics that are currently being measured. Even then, not all contractors measure them systematically. Furthermore, some software products that are generated with the SEE, such as documentation products, are not represented by these metrics. Nor do the metrics capture some of the important features of the SEE, such as how fast updates to the application software can be turned around, or how fast modifications to the SEE itself can be made.

• User support is very important, and every one of the evaluated SEEs offered training courses. One SEE went even further in their user support by providing each project using it with dedicated personnel whose job is to assist the users in their daily use of the SEE and at their request.

• Strong barriers do exist, however, between the SEE developers and the SEE users. The SEE developers tend to come from different organizations than the SEE users. Furthermore, little effort is expended towards soliciting inputs from the users. As a result, a sales job must be done to encourage the user community to adopt the SEE. However, training courses are provided for all the evaluated SEEs, as mentioned above.

• The investment of large sums of money into developing a SEE puts pressure on the contractor's management to capitalize on their investment by pushing the use of the SEE very hard. In order to avoid the situation in which a given SEE is being used on a project only because the company wants to amortize the cost of developing the SEE, a careful SEE evaluation must be performed to evaluate the SEE and determine its adequacy for the project.

5.2. Comparison of SEE Features

The following descriptions of the evaluated SEEs have undoubtedly changed since the time of the evaluations. They provide a snapshot of the evaluated SEEs, taken at the time of the evaluations, i.e, between November 1989 and May 1990.

Each of the four SEEs has a different underlying architecture, which determines how and to what degree the software engineering process can be supported. The features selected below are those SEE components and SEE attributes which best summarize the support that can be expected from the evaluated SEEs.

- Platform and Operating System

SEE1:	SUN/UNIX
SEE2:	VAXstation/VMS
SEE3:	SUN/UNIX
SEE4:	VAX/VMS

- Target

SEE1:	Same as host
SEE2:	Same as host
SEE3:	Project-dependent, connected over Ethernet with TCP/IP (Transmission Control Protocol/Internet Protocol)
SEE4:	Project-dependent, connected over Ethernet with TCP/IP protocol

- Use of Repository

SEE1:	No
SEE2:	Yes
SEE3:	Yes
SEE4:	No

- Tool strengths

  SEE1: Ada Prototyping
  SEE2: Program Development Folder (PDF) Tool (both a tool for generating PDFs and an interface to other tools)
  SEE3: Requirements Traceability
  SEE4: Ada Design

- Tool weaknesses

  SEE1: Testing, Configuration Management, Project Management
  SEE2: Testing, Project Management
  SEE3: Testing
  SEE4: Testing

- COTS[4] versus IH[5] tool development

  SEE1: Primarily IH
  Design tools, which constitute the majority of the tools, were developed IH. The user interface is from Cadre's Teamwork. The Ada compiler tools are from Verdix.

  SEE2: Primarily IH
  The design tools and the user interface are IH developments. The compiler tools are from DEC. The majority of the remaining tools are IH developments.

  SEE3: Primarily COTS
  Requirements Analysis and Requirements Traceability are IH developments. The Project Management Database is an IH development using COTS called Sybase. Design tools are COTS and include Adagen for Ada. Compiler tools are COTS and include Verdix for Ada.

  SEE4: Primarily IH
  Requirements, Design and Configuration Management tools are all IH developments. Compiler tools are COTS and are project dependent.

[4] Commercial Off The Shelf
[5] In House

- Common User Interface

SEE1:	Yes
SEE2:	Yes
SEE3:	Yes
SEE4:	No

- Data Interface Standards

SEE1:	None
SEE2:	None
SEE3:	None
SEE4:	None

- Use of Task Management Services

SEE1:	Minimal
SEE2:	Extensive
SEE3:	Unclear from available documentation
SEE4:	Unclear from available documentation

- Tailorability

SEE1:	Weak
SEE2:	Weak
SEE3:	Weak
SEE4:	Weak

- Software Development Process Supported

SEE1:	2167A
SEE2:	2167A
SEE3:	2167A
SEE4:	2167A

- Design Methodologies Supported

SEE1:	Object Oriented Prototyping[6]
SEE2:	Structured Analysis and Structured Design
SEE3:	Booch and Buhr Notation[7]
SEE4:	Ada Real-Time Design Methodology[8]

---

[6] In fact use of SEE1 requires adherence to this methodology, which was defined by Steve Mellor.

[7] An object-oriented Ada design notation defined by Grady Booch and later refined by Ray Buhr.

[8] A methodology for Ada Real Time design, developed by Kjell Nielsen and Ken Shumate.

- Usage on DOD Projects

  SEE1: Partially used on one project but no usage data were collected
  SEE2: Partially used on one project, some data collected (LOC/MM and Detected Errors/MM)
  SEE3: Used on several projects, some data collected (Time to completion, measured for 27 different software life-cycle activities)
  SEE4: Used on several projects, some data collected (LOC/MM and Detected Errors/MM)

- Reuse Repository

  SEE1: No
  SEE2: No
  SEE3: Yes
  SEE4: No

6. EVALUATION PROCESS FINDINGS

Several lessons were learned about the process of evaluating contractor SEEs. These findings will be used to refine the evaluation approach defined for this task into a SEE evaluation methodology. Before the evaluations were started, an evaluation approach was formulated. The evaluation approach consisted of evaluating the SEEs along three different dimensions: the functionality of the SEE, its architecture, and its usage. The SEE functionality consists of the different software engineering activities that the SEE automates. The SEE architecture, refers to the components of the SEE, how well they fit together, what process is followed in acquiring them and maintaining them, and so forth. The SEE usage is concerned with determining whether the SEE is being used in an optimal way, how well the users are trained, how their inputs are solicited and, most importantly, how the gains obtained from using the SEE are assessed.

The information used to evaluate the SEEs was acquired from the contractors either through site visits, or from documents when site visits could not be arranged. Site visits were favored because they allowed the evaluators to get direct answers to their questions by interviewing contractor personnel, and because they enabled the evaluators to witness a demonstration of the SEE. Thus, site visits consisted of demonstrations of the SEE and interviews with contractor personnel. The information obtained from the contractors was used to fill out a SEE questionnaire, which is included in the Appendix.

A SEE demonstration, if it is not carefully planned with the evaluators, will usually turn into a sales pitch by the contractor. Therefore the evaluators should plan the contents of a demonstration, with the contractor, prior to the site visit. In order to plan for an efficient SEE demonstration, the evaluators must know the components of the SEE. The SEE documentation should therefore be acquired by the evaluators prior to the demonstration. One way to capture the SEE documentation is to have the contractor fill out the questionnaire prior to the site visit. The questionnaire can be updated by the evaluator at the end of the evaluation.

Interviews with contractor personnel are most valuable when conducted with people representing the different perspectives of a SEE. The SEE architecture and components are best described by the SEE developers and the process of building the SEE by the developers' management. However, the usage of the SEE is best evaluated by talking to SEE users, and the SEE gains should be discussed with managers from the projects that have used the SEE. Interviews should be conducted with SEE developers, SEE users, and with managers from the respective organizations.

The questionnaire was found to be very useful at covering a large number of SEE issues. A general questionnaire, however, does not apply equally well to different situations. If the SEE evaluation is performed for a given software development project, it is recommended that the questionnaire be extended for the specific needs of the project, such as security requirements, methodologies required for the project, and so forth.

The information obtained from the evaluated SEEs about their functionality and their architecture was fairly adequate. However, a complete assessment of the usage aspect of the evaluated SEEs was not possible from the available usage information. The collection of SEE usage metrics by DOD contractors is insufficient. In general, there does not appear to be any consistent effort to monitor the usage of the SEEs and to systematically collect data regarding the gains obtained from using the SEE, nor is there any agreed upon set of SEE metrics. There is a need for the definition of an extensive set of SEE metrics and for standardizing their use by contractors.

7. CONCLUSION

This study was based on a survey of ten DOD contractors of which four SEEs were evaluated. The results of the evaluations show that SEE development is an expensive and lengthy process, and that the cost of a SEE cannot be amortized on one project only. The government currently is addressing the problem of SEE cost through the STARS program which will provide a framework for using commercial tools and reusable components. The study also shows that SEEs come in many different varieties and architectures which makes SEE evaluations difficult. The standard, DOD-STD-2167A, requires the use of a SEE, yet it does not provide any guidelines for selecting the optimal SEE for a given project. A SEE evaluation methodology must be developed and used in conjunction with DOD-STD-2167A.

The study finds that it is not clear how to assess the gains obtained from a SEE, especially when they are traded off against the cost of developing it. One activity that must be included in any benefit assessment is the analysis of usage data. Currently, contractors are not collecting adequate usage data . Even if they wanted to, they may not know what kind of data are needed for a complete assessment of the benefits obtained from using their SEEs. SEE usage metrics need to be defined and standardized, and contractors need to be encouraged to measure and collect them.

As more and more SEEs are used in the DOD community, SEE evaluations will play an important role, both in proposal evaluations and in contract monitoring. The results from this task will be used to formulate a methodology for evaluating and monitoring SEE developments and SEE usage. A follow-up to this study is planned. Any other DOD contractors who wish to share information about their SEEs with the authors are strongly encouraged to do so by contacting the Computer Resources Management and Standards Office at The Aerospace Corporation.

APPENDIX: SEE QUESTIONNAIRE

A. SEE Definition

1. Do you have a SEE?

2. What was your motivation in defining your SEE?

3. What processes does your SEE support?

4. Methods:
   a. Are any particular methods assumed?
   b. Do you use prototyping for designing critical performance software elements?
   c. Do you use prototyping to design critical elements of man-machine interfaces?
   d. Do you develop re-usable code, if so how is it supported by your SEE?

5. What languages does your SEE support?

6. What are its software components (See Table 1.)?

7. What is the function of each tool?

8. What languages are they written in (give the percentage in each language)?

9. What is the origin of each tool (vendor versus in-house)?

10. How much automation do you have in the area of document generation (See Table 1.)?

11. How are the different tools integrated?

12. Does the environment contain a repository? If so, what is its structure?

13. What are the hardware components?

    Host computers
    Target computers
    Operating Systems
    Terminals
    Network
    Other

14. Is there more than one environment?

B. History of SEE

1. How long has the SEE been in place?

2. What projects have used it and for how long?

3. Are there data on the productivity impact from using the SEE?

4. What is the nature of the applications using the SEE (real-time system, data processing, signal processing, control, telemetry, simulation, and so forth)?

5. Does your SEE support host/target applications?

C. Visibility into the SEE

1. Is there literature documenting your SEE?

2. Are you willing to give us a demo?

3. How much would we see in the demo?

4. Are there contact points that are:
   a. Tool gurus
   b. Technical users
   c. Project manager users?

D. SEE Usage

1. How widespread is the usage of your SEE within your company?

2. How strong is your company's commitment to your SEE?

3. Is there any resistance to using your SEE?

4. How are the users trained?

5. How is your SEE promoted?

E. Experience and Plans

1. What benefits have you realized by using your SEE?

2. What have been the pitfalls?

3. What has been your experience with the tool vendors?

4. How did you choose the tools you have?

5. How expandable is your SEE?

6. What are your future plans?

F. Configuration Management

1. What tools are used for Configuration Management of the SEE? Are they the same as the Configuration Management tools offered by your SEE?

2. How are the COTS tools upgraded and tested for acceptance?

3. *In-House developed tools:*
   a. Are different versions of the tools maintained for different projects?
   b. Is there a Software Development Library?

G. Cost and Accessibility

1. How is/will the cost of the SEE be amortized?

2. Will the SEE be delivered to the customer for use in the software support environment, and if so, under what conditions?

Table 1. SEE Software Components by Functionality

1. General support tools

   - Operating systems
   - Editors
   - Document production tools

2. Specification tools (requirements and design)

   - Graphical representation
   - Symbolic representation (Program Design Language, for example)
   - Specification Analysis
   - Traceability
   - Automatic document production

3. Implementation tools

   - Automatic code generators
   - Assemblers
   - Compilers
   - Linkers
   - Debuggers
   - Code analyzers

4. Testing tools

   - Test generators
   - Data generators
   - Test coverage analyzers
   - Traceability

5. Integration tools

   - Simulators
   - Loaders (for host/target systems)

6. Analysis tools

   - Performance analyzers
   - Throughput analyzers
   - Data gatherers
   - Data reducers

7. Evolution tools

   - Configuration Management
   - Change analyzers
   - Reverse Engineering

8. Management tools

   - Risk assessment and management (Pert Charts, and so forth)
   - Schedule management
   - Software Status Reporting
   - Productivity metrics

REFERENCES

1. Humphrey, W.S., "A Method for Assessing the Software Engineering Capability of Contractors," CMU/SEI-87-TR-23, 1987.

2. Petersen, G., Van Buren, J., and Nilson, S., "Interim Report on Requirements Analysis and Design Tools," Software Technology Support Center, 1900.

3. Byrnes, C.M., "Computer-Aided Software Engineering Environment and Tool Evaluation," MITRE WP-28542, 1990.

4. Byrnes, C.M., "Requirements Driven Design Evaluation," MITRE WP-28380, 1989.